# NAVAL POSTGRADUATE SCHOOL
# MONTEREY, CALIFORNIA

# THESIS

## REAL-TIME INTRUSION DETECTION FOR WINDOWS NT BASED ON NAVY IT-21 AUDIT POLICY

by

H Steven Kremer

September 1999

Thesis Advisor: Neil Rowe
Co-Advisor Ronald Broersma

**Approved for public release; distribution is unlimited.**

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*

| | | |
|---|---|---|
| **1. AGENCY USE ONLY** ***(Leave blank)*** | **2. REPORT DATE**<br>September 1999 | **3. REPORT TYPE AND DATES COVERED**<br>Master's Thesis |

| | |
|---|---|
| **4. TITLE AND SUBTITLE**<br>REAL-TIME INTRUSION DETECTION FOR WINDOWS NT BASED ON NAVY IT-21 AUDIT POLICY | **5. FUNDING NUMBERS** |
| **6. AUTHOR(S)**<br>Kremer, H Steven | |
| **7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**<br>Naval Postgraduate School<br>Monterey, CA 93943-5000 | **8. PERFORMING ORGANIZATION REPORT NUMBER** |
| **9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)** | **10. SPONSORING / MONITORING AGENCY REPORT NUMBER** |

**11. SUPPLEMENTARY NOTES**

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

| | |
|---|---|
| **12a. DISTRIBUTION / AVAILABILITY STATEMENT**<br>Is approved for public release; distribution unlimited. | **12b. DISTRIBUTION CODE** |

**13. ABSTRACT** ***(maximum 200 words)***

A Navy directive orders the migration of Navy computer systems to an Internet-connected network of Windows NT workstations and servers. Windows NT possesses the security features of a class C2 computer system but does not offer a standard real-time host-based tool to process the security-event audit data to detect intrusions or misuse. We discuss what would entail in general. We also report on experiments with a sensor program, which resides on each workstation and server in the network and provides some real-time processing of NT host-based events. It passes information to an Agent that communicates to other Agents in the network, in an effort to identify and respond to an intrusion into the network. The Navy audit policy and the methods of implementing the policy are also investigated in this thesis.

| | |
|---|---|
| **14. SUBJECT TERMS**<br>Intrusion Detection, Artificial Intelligence, Autonomous Agents, Computer Security | **15. NUMBER OF PAGES** |
| | **16. PRICE CODE** |

| | | | |
|---|---|---|---|
| **17. SECURITY CLASSIFICATION OF REPORT**<br>Unclassified | **18. SECURITY CLASSIFICATION OF THIS PAGE**<br>Unclassified | **19. SECURITY CLASSIFICATION OF ABSTRACT**<br>Unclassified | **20. LIMITATION OF ABSTRACT**<br>UL |

# REAL-TIME INTRUSION DETECTION FOR WINDOWS NT BASED ON NAVY IT-21 AUDIT POLICY

H Steven Kremer
B.S.degree, San Diego State University, 1982

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN SOFTWARE ENGINEERING**

from the

**NAVAL POSTGRADUATE SCHOOL**
**September 1999**

Author: H Steven Kremer
H Steven Kremer

Approved by: Neil C. Rowe
Neil Rowe, Thesis Advisor

Ronald Broersma, Co-Advisor

Dan C Boger
Dan Boger, Acting Chairman
Computer Science Department

# ABSTRACT

A Navy directive orders the migration of Navy computer systems to an Internet-connected network of Windows NT workstations and servers. Windows NT possesses the security features of a class C2 computer system but does not offer a standard real-time host-based tool to process the security-event audit data to detect intrusions or misuse. We discuss what would entail in general. We also report on experiments with a sensor program, which resides on each workstation and server in the network and provides some real-time processing of NT host-based events. It passes information to an Agent that communicates to other Agents in the network, in an effort to identify and respond to an intrusion into the network. The Navy audit policy and the methods of implementing the policy are also investigated in this thesis.

## TABLE OF CONTENTS

# ACKNOWLEDGEMENTS

I would like to thank Professor Neil Rowe of the Naval Postgraduate School for all his effort in providing direction and support for this effort. Also his efforts in providing logistic support in getting the proper routing of documents and the signatures needed in order to get this thesis approved in time for September 1999 graduation.

# I. INTRODUCTION

## A. SYSTEM VULNERABILITIES

The initial Internet protocols were not designed with security in mind but with simplicity of connectivity. The rapid growth of the Internet has created a vast opportunity for business and communication world-wide, but it has also opened up a playground for those who would like to use their knowledge against vulnerabilities in the computer networks attached to the Internet. Their exploits make the paper daily, where we hear of their ability to use computers to steal money and services, disrupt networks, commit espionage, and compromise sensitive information.

Military networks have long been the prime targets to those who would take advantage of network software and operating-system vulnerabilities to modify, steal, inappropriately disclose, and destroy sensitive military data. The misuse of government data has been largely the result of users who have authorized access to a computer. Ineffective access controls to the data have resulted in the compromise of that data. The actions of those who would inappropriately access military computers and compromise sensitive data potentially put the security of the world at risk.

## B. INTRUSION AND MISUSE DETECTION

When inappropriate usage, whether intrusion or a misuse, occurs within a computer system, the ideal response to the event would be an alert and initiation of counter measures. The best response would occur as soon as the event was detected. Events that would trigger a response would be those defined as prohibited in the organization's security policy. If the event is audited within the compute, then the capability to automate the detection and analyze and report on the event is required. A real-time detection method based on the security policy of an organization certainly is an advantage over a system administrator manually viewing the audit log data after misuse has occurred. The audit logs can contain large amounts of data and viewing them manually is tedious and prone to error. While real-time detection would assist the local environment, information on intrusion or misuse should also be shared with the network environment to allow a network response if appropriate.

The network environment encompasses the host-based environment and usually the same security policy applies. The advent of the networked file servers and print servers at the least makes the sharing of information more needed and more justified. An environment of distributed sensors gathering relevant audit data, and distributed agents capable of analyzing that data in both a host-based and a network environment, could provide for an immediately responsive distributed system of security.

## C. SUMMARY OF CHAPTERS

This paper will address using the security policy of an organization to implement a system audit policy with a real-time intrusion and misuse detection system. A sample program which will act as sensor for a larger system will be demonstrated.

Chapter II discusses the current Navy directives that specify Windows NT network environments, their security configurations, and policy.

Chapter III discusses the implementation and adaptation of the Navy security policy in respect to security event log (audit) policy using a third party security product.

Chapter IV discusses the NT Audit Sensor, a real-time sensor that uses the NT security event log to provide security-event data to an Agent who analyzes the data and provides communication to the entire network. The design and the execution results of the sensor program are discussed.

Chapter V discusses the results of this approach and how it meets the requirements of a host-based intrusion-detection system.

The Appendix contains the source code for the NT Audit Sensor program.

# II. APPLICATION DOMAIN

## A. THE NAVY IT-21 DIRECTIVE

The Navy has issued an order (R 300944 MAR 97, INFORMATION TECHNOLOGY FOR THE 21$^{ST}$ CENTURY) [Ref. 3] providing direction for migration to the Microsoft Windows NT 4.0 operating system for workstations and network servers, to be completed by December 1999. With this beginning message and subsequent additions, Information Technology for the 21$^{st}$ Century (IT-21) became the Navy directive specifying the transition of command, control, communications, computers and intelligence programs of record to a NT workstation-based network. The Windows NT environment is composed of the following items: NT workstations, NT servers (file, print server), and NT primary domain and backup domain controller.

## B. C2-LEVEL SECURITY REQUIRES AUDITING

Windows NT was designed to be a general server operating system, offering application, communication, and file and printer sharing capabilities. Windows NT servers and workstations have a common security architecture and a similar set of security features allowing for functionality that is consistent across both platforms. So the auditing of system events is similar on servers and workstations.

In Windows NT, every protected object is represented by the same type of structure (the same object abstraction), and the access determination to an object uses the same access control mechanism (the security reference monitor). Users are identified and authenticated before any accesses to system resources are granted. Subsequent accesses to resources are controlled using Access Control Lists that allow the owner to specify exactly who and what type of access to the object is allowed. Access checks are done for system resource calls on one reference monitor [Ref. 21]. This provides for a highly reliable access-control mechanism.

Security in Windows NT is based on guidelines developed by the U. S. Department of Defense. The requirements for A-, B-, C-, and D-level secure products are outlined in the Department of Defense Trusted Computer System Evaluation Criteria (TCSEC) published by the

National Computer Security Center [Ref. 8]. C2 is defined by a set of policies about how a particular kind of secure system operates. Auditing is one of the most important security requirements for a C2-level system. To meet the auditing requirement, the operating system must be able to monitor attempts to access specific system and network objects and attempts to modify the privilege of user accounts. It must also monitor attempts to bypass security mechanisms and change auditing policies. Specific fields must be contained in the event record to meet C2-level security requirements.

The purpose of auditing is to generate data that will allow the detection of computer misuses. Of particular importance are events that typical firewalls and other network-based detectors cannot catch, such as adding trusted domains, adding administrative accounts, and perusing sensitive files. Brinkley [Ref. 2] categorizes computer misuse into four broad areas: theft of computer resources primarily by unauthorized users, disruption of computational services (an example of this is denial of service), unauthorized information disclosure (i.e. espionage), and unauthorized information modification (an example of this is modification of accounting data). Cohen [Ref. 4] creates a classification scheme by defining close to a hundred classes of methods of attacks. Kosoresow [Ref. 10] discusses using system-trace calls when generating attack scripts to determine abnormal program execution. Using this method attacks can be matched to other classes of attacks; this may provide developers with the information necessary to counterattack for an entire class of attacks.

Historically, auditing has always been done on a per-machine basis, recording specific operating system activities in a secure place for later analysis. Little thought was given to the redesign of auditing subsystems when networking was introduced. Lunt [Ref. 11] discusses the need for automated tools. But to obtain the most information from auditing, the data from different machines must be compared to spot trends and behaviors that may be occurring between machines. This analysis could yield patterns that could be considered security threats to the computer systems and/or network. Esmaili [Ref. 12] discusses translating a sequence of audit records into a sequence of action classes and using an accumulated risk-factor ranking to determine an action. Puketza [Ref. 19] discusses an approach to intrusion detection by creating intrusion signatures, encapsulations of the identifying characteristics of specific intrusion techniques, and a pattern-matching mechanism that searches for the intrusion signature in user audit logs.

Auditing requires a tradeoff between level of security, system overhead, and complexity of the analysis of the audit trail. In many cases administrators audit far too little because the high system overhead and complexity of the analysis overshadow the security benefits. In order to achieve higher security through auditing, a program that generates relevant data and minimizes system overhead consistently across the enterprise must be implemented.

## C. WINDOWS NT AUDITING

The Windows NT audit subsystem can collect event data on a wide range of system and operation events, but the NT auditing has largely been ignored by users. This is because NT provides neither an enterprise-wide mechanism for setting up and distributing audit policies nor the tools needed for analysis and reporting on the audit events.

Up until NT Service Pack 4.0, it was impossible with the NT supplied tools to realize an effective enterprise-wide audit-based security policy. Using the standard NT-based Explorer software to set up access control on individual files and directories on each host computer could lead to significant errors and omissions across an enterprise.

The Security Configuration Manager (SCM), a component of NT Service Pack 4 [Ref. 17], permits users to define a template, or allows use of pre-defined templates that contain desired security settings. These settings are used to implement secure NT systems in the areas of auditing, user rights, account policies, C2 configuration, restricted group membership, file system and registry access control lists, and most importantly, security options. The security options are nothing more than lists of features that correlate directly to registry-key settings. In previous versions, the only way to configure the security options was to manually manipulate the registry or use the C2 Configuration Manager tool from the NT Resource Kit.

The Security Configuration Editor, also new with NT Service Pack 4, provides a simplified user interface for defining and installing security templates domain-wide. These templates define security in six areas: system security policy, user accounts, rights and privileges, directory and file system security, registry security, and system services [Ref. 22].

The only Microsoft tool available for audit reporting is the Windows NT system application the Event Viewer, and it provides only the most basic of features, the ability to view each event record. This involves a huge amount of data so that human perusal of audit data is all but impossible across the enterprise. Auditing management tools is an area that the developers of NT have left to the third-party developers.

## D. ENTERPRISE-WIDE SECURITY-AUDIT POLICY

Any security product for host-based or enterprise-security auditing requires that we establish a relevant enterprise-wide security audit policy, and distribute it to all computers in the enterprise. The audit policy specifies which operating-system events and file operations will be logged in the system-event logs. The implementation of this policy is critical for intrusion detection and behavioral analysis.

An initial attempt to define a security-audit policy for the Windows NT environment is outlined in the IT-21 document [Ref. 6 & 7]. This document gives post-installation instructions for a C2 configuration, auditing, securing the registry, managing the file system, creating system policies and user profiles, controlling user accounts and rights, maintaining system repair data, and installing current NT service packs which provide software fixes for known problems. This document delineates the Navy standard NT-based audit specification for use on all Navy networked NT workstations and servers. Because of NT's limited audit-management facilities, the document authors were apparently convinced that a directory, file, and registry audit policy would not be feasible, and did not include it in the IT-21 specification.

## E. EVENT-LOG ANALYSIS AND REPORTING

After establishing and implementing a security-audit policy, the audit data must be managed. The event logs must be analyzed, and used to generate reports.

The usefulness of event-log auditing depends upon the accuracy of the information in the log and on the timeliness of the response to it by a system administrator. The event logs are usually analyzed after the actual or attempted intrusion or misuse is over. This type of security analysis, offline analysis, is typically well after the fact. But with a real-time element, audit analysis can become a proactive security monitor and not just a forensic tool after a security threat has occurred.

So audit data must be evaluated in real-time, with appropriate alarm warnings and responses provided. Automation of this can provide immediate security benefits to the host-based computer, and if warnings are communicated to all the computers in the network, an appropriate network response can be initiated. Staniford-Chen [Ref. 20] discusses the need for standardizing formats, protocols, and architectures to allow the sharing of intrusion data across a network, making the data available to a variety of security programs. This also includes the conversion of audit trails into a standard format to permit cross-host analysis.

Some of the current intrusion-detection systems are network-based and attempt to detect attacks by examining the contents of transmitted packets, watching for intrusion signatures and patterns of activity that do not fit the normal user pattern. Examples of network-based intrusion detection security products are Internet Security Systems RealSecure, Axent Technologies Intruder Alert, and Cybersafe Centrax. A host-based real-time intrusion-detection alert mechanism does not preclude the need for network-based intrusion-detection mechanisms but rather augments its coverage. Most of the same developers of network-based intrusion detection systems provide host-based intrusion-detection products including Internet Security Systems System Scanner, Axent NetProwler, and CyberSafe Centrax.

## F. ESTABLISHING THE NT AUDIT POLICY

The NT System Administrator can customize security auditing through the NT administrative tools: User Manager for Domains (for servers) and User Manager (for workstations). The choices the administrator has for auditing concern success and failure on event categories: Logon and Logoff, File and Object Access, Use of User Rights, User and Group Management, Security Policy Changes, Restart, Shutdown, and System and Process Tracking. Login/logoff (success and failure) can be used at both the local and at the network level to detect intrusion attempts. File and object (failure) can be used to examine attempts to read, execute or modify critical system files and sensitive information. The system administrator can designate specific files and directories if File and Object Access is being audited; the types of events (Read, Write, Execute, Delete, Change Permissions, and Take Ownership) can be specified for users and groups. Use of User Rights (success and failure) provides for auditing the attempted use of special privileges. User and Group Management (success and failure) auditing checks changes to the accounts database, such as the addition of users. Security-policy changes can be reviewed for discrepencies against the local security policy. Restart, Shutdown, and

System refer to system events such as restarting or shutting down the system and attempting to change the audit policy. Process Tracking tracks the creation, destruction, and access of process objects.

The System Administrator can also audit specific registry keys. The registry is a repository of NT system and user configuration information and its event types include Query Value, Set Value, Create Subkey, Enumerate Subkeys, Notify, Create Link, Delete, Write Discretionary Access Control, and Read Control. The system administrator can also audit printers for the events: Print, Full Control, Delete, Change Permissions, and Take Ownership for groups or users.

Audit policy specification tools provided by third-party developers offer NT system administrators a way to review the auditing policy, file and directory auditing settings, and registry auditing settings. They also facilitate a consistent and reproducible series of settings that can be easily applied to multiple hosts.

## G. IT-21 SECURITY AUDIT POLICY

The current IT-21 audit policy [Ref. 7] recommends that system auditing at least cover:

- Login/logoff (success and failure)
- File and object (failure)
- User and group management (success and failure)
- Security policy changes (success and failure)
- Restart shutdown and system (success and failure)
- Specific directory and file auditing.

However, third-party software can go considerably beyond these.

# III. AUDITING NT SYSTEMS IN COMPLIANCE WITH THE IT-21 DIRECTIVE

## A. IMPLEMENTING AN ESTABLISHED SECURITY AUDIT POLICY

We now discuss an implementation of the IT-21 audit policy conducted during an evaluation of the CyberSafe-produced Centrax security-products by SPAWAR Code D871 employees. We will propose auditing extensions that should be considered for heightened NT security. The auditing-policy recommendations are made without regard to the difficulty of achieving them; in other words they are "best-practices" recommendations for security, but not for cost or complexity. Implementation of the IT-21 security audit policy will ensure that most data about potential security threats is recorded, but IT-21 does not include a policy for examining and reacting to potential threats.

The only Microsoft tool provided with NT for analyzing the NT event logs is the NT Event Viewer. Its main deficiencies are:

- The inability to automatically notify an administrator or otherwise respond when an event occurs.
- The inability to detect and respond to a group of events, known as a signature.
- The inability to analyze and view an aggregate of event logs from multiple machines.
- The difficulty in understanding the Event Details. Often many events comprise an operating system action, and the administrator must sift through many entries to infer it.
- The lack of query/reporting tools for data analysis.

With the installation of NT Service Pack 4.0, the new Security Configuration Manager (SCM) utility permits users to define a template or use predefined templates that contain desired security settings. These settings are used in the areas of auditing, user rights, account policies, C2 configuration, restricted group membership, file system and registry access control lists, and most importantly, security options (lists of features that correlate directly to registry-key value settings).

To use the audit logs more effectively within a IT-21 audit policy, an audit subsystem with the following characteristics is desirable:

- Full audit-policy creation and deployment from central management facility.
- Centralized storage and management of enterprise log data.
- Effective, easy, and centralized analysis and reporting using the enterprise-wide audit data of all targeted computers, for computer misuse detection and long-term behavioral analysis.
- Real-time analysis of log data along with administrator notification and automatic response when predefined data (signatures) is logged.

To explore security-audit policy in the NT environment, the IT-21 audit policy requirements were implemented using the Centrax security software at SPAWAR. Cybersafe personnel established an initial audit policy that met the requirements in the IT-21 specification [Ref. 5]. During the Centrax testing at SPAWAR, the audit policy further in conformance with SPAWAR enterprise-security policy.

## B. IT-21 AUDIT POLICY AND CENTRAX INITIAL AUDIT POLICY RECOMMENDATIONS

A significant difference between It-21 and Centrax audit policy is in successful file and object access. Auditing on critical files can reveal much about user behavior, especially user behavior allowed by the security policy. A note in IT-21 [Ref. 6] that states that file and object access (success) auditing will "fill the event logs rapidly and should not be enabled." This is only true if the administrator specifically sets auditing on a large number of files. Setting file and object (success) auditing will not, in itself, cause the event log to fill.

The following table summarizes the auditing requirements of IT-21 and the Centrax recommendations for extensions to those requirements.

| IT-21 Recommendation | Centrax Recommended Extensions |
|---|---|
| System auditing: login/logoff (success and failure)<br>File and object (failure)<br>User and group management (success and failure)<br>Security policy changes (success and failure)<br>Restart shutdown and system (success and failure) | File and object access be audited on success. |
| File auditing: executables of NT operating system (C:\winnt) as a minimum should be audited. | There should be more file auditing, especially in the system areas of the file system. |
| Directory auditing: system repair area (C:\winnt\repair) and system configuration area (C:\winnt\system32\config) minimum should be audited. | There should be more directory auditing, especially in system, critical-application, and critical-data areas of the file system. |
| Registry key auditing: auditing not recommended. | Certain keys should be audited only on critical file and print servers. |
| Audit Log File Permissions should be audited. | Auditing should detect tampering with this file. |
| Setting permissions on archive directory | Auditing should detect tampering with files in archive directory. |
| Creating application, system and security log archive should be audited. | No change. |

Table 1. IT-21 Audit Policy and Centrax Recommended Extensions [Ref. 5]

IT-21 only suggests auditing for all executable files in the C:\Winnt directory, and further auditing for "particular files on your system." In general, only this minimum suggestion will be followed by most organizations because individual file auditing is difficult to set up, especially across the enterprise, using the standard NT tools. However, we see good reasons to audit substantially more, including the benefits of trend and behavior analysis on audit logs. The following general file and registry-key audits provided by Centrax [Ref. 5] should be considered on any system.

| File/Directory/Registry Key | Type | Comment |
|---|---|---|
| *.*.exe | WD | Executable files in system space |
| *.*.bat | WD | Batch files |
| *.*.com | WD | Command files |
| *.*.dll | WXD | Dynamic link libraries should be put here |
| *.*.ini | WD | Initialization files including boot.ini |
| *.\bootsect.dos | WD | Various operating system files |
| *.\ntldr | WD | Various operating system files |
| *.\ntboot.sec | WD | Various operating system files |
| *.\winnt*.exe | WD | Various operating system files |
| *.\winnt*.com | WD | Various operating system files |
| *.\winnt*.bat | WD | Various operating system files |
| *.\winnt*.dll | WD | Various operating system files |
| *.\winnt*.sys | WD | Various operating system files |
| *.\winnt*.ini | WD | Various operating system files except \winnt\profiles\*.ini |
| *.\winnt\system32\config\*.evt | RWD | Event logs |
| *.\winnt\repair\ | WD | System repair directory |
| *.\winnt\Program Files\ | WD | Installed programs |
| *.\temp\ | D | Temporary use by many applications |
| HKLM\SYSTEM\CurrentControlSet\Services\EventLog\[Log Name] | WD | Registry Keys. |

Table 2. Centrax File And Registry-Key Audit Recommendations [Ref. 5]

| | | |
|---|---|---|
| Legend: | * | = Wildcard symbol |
| | HKLM | = The local-machine NT registry |
| Audit Types: | R | = Read-Access logging |
| | W | = Write-Access logging |
| | X | = Execute-Access logging |
| | D | = Delete-Access logging |

The Windows NT Registry is a database used to store critical data about a system's software, hardware, and security. Registry keys are like file system directories, not files. A key can have multiple values which in turn contain data like a file does. Registry key values like those of a key such as HKLM\SYSTEM\CurrentControlSet\Control\Lsa, contain values pertaining to the Local Security Authority and cannot be effectively audited. The question then becomes whether or not to audit the key, which affects all of the values associated with the key.

Because of this limitation, registry-key auditing is not nearly as granular as it needs to be for detecting misuse. This is an area in the NT auditing subsystem that needs enhancing.

Many programs open the registry for a write even though the intent is simply to read. If a key is opened for a write, it is audited as such, even if no changes were made to the registry key. For this reason, auditing the registry can generate many false alarms. So auditing registry keys should be done for those keys which users should not access, such as the audit-log keys and others which are directly manipulated by the operating system.

Each of the event logs must be archived to prevent overwriting of data. Furthermore, permissions must be set to prevent unauthorized access to the archived logs. Lastly and most importantly, log files should be audited for any access to see if they have been tampered with.

## C. CENTRAX PRODUCT ARCHITECTURE

Centrax is a security product that integrates host- and network-based intrusion detection and vulnerability assessment into one package driven by a single Command Console. A centralized policy management facility enables the administrator to setup and deploy comprehensive audit policies and the corresponding host-based intrusion-detection and network intrusion-detection policies.

The architecture for Centrax uses a Console-Agent technology, with the central Command Console providing the majority of the processing. The agents that run on the monitored computers are very simple, in most cases only “waking up” periodically to send logs to the Command Console for analysis. Exception to this are the real-time host and network agents, which process event logs “on the fly” and support a variety of network or host-based alerts by methods such as email and pagers.

Three system logs (Application, System, and Security) for all clients, along with the network data, are stored in a database on the central Command Console. The logs can be analyzed and reported on, for both computer misuse detection and long-term behavioral analysis using statistical methods. The combined use of host-based and network-based intrusion detection, along with the ability to correlate the data from both detection methods, provides a complete intrusion-detection package.

### 1. Audit Policy Manager

The Centrax Audit Policy Manager is a utility for specifying audit policies. A graphical user interface is provided for defining the general policy and for setting up file/directory and registry-key audits. The administrator can choose from predefined audit policies or can create policies with custom settings. Specifications can be applied immediately or saved for later.

### 2. Network-Based Probe

The Centrax network probe is software used to capture network packets for analysis.

### 3. Vulnerability Assessments

The configuration of all the computers in the network can be assessed from the Command Console for their overall security posture including privileged-accounts information, required NT File System (NTFS) drive configuration, secure login configuration, password policy, screen saver status, and overall system configuration.

### 4. Host-Based Audit Policy

Centrax provides a utility for creation and modification of audit-policy system parameters, system files, system folders, and system-registry keys. It provides this from a central Command Console and permits the data to be distributed across the network.

### 5. Alert Methods and Responses

Critical-event notifications, triggered by individual activity signatures, can be conveyed by automated pager or e-mail. An automated response can be to log off the user, disable the local account, or shut down the target computer.

### 6. Real Time Host-Based Critical Signatures For Windows NT

Centrax recommended that only computer or server-specific critical signatures be monitored in real-time. The Windows NT 4.0 attack signatures covered by the Centrax product were [Ref. 5]:

- The reading, modification, and accessing of the critical system registry.
- The reading of the critical project registry key.
- The reading of a registry key that has been designated as a decoy.
- The reading of the Security Product registry key.
- Modification, addition, or deletion from these groups: account operators group, administrators group, backup operators, domain administration, print operators, group membership server operators, and guests.
- The execution of critical programs.
- Access or attempted access to an audited file.

## D. CENTRAX TESTING

Centrax was initially installed on 13 Windows NT computers with various hardware and operating systems in the SPAWARSYSCEN SAN DIEGO D87 laboratory and offices; it was later expanded to incorporate approximately 50 computers throughout the building. Data was collected for several months to determine what auditing condition could generate sufficiently detailed data to be useful without overwhelming available storage capacity. The collected data was analyzed using Centrax's statistical analysis capabilities.

Certain "misuse" events and simulated attacks were done to test whether they could be detected by Centrax software. These included three consecutive login failures with different types of accounts, changing and restoring the audit policy, installing programs with administrative privileges, creating an administrator account, clearing the security log, editing, taking ownership and renaming audited files, and executing privileged programs. The event times were recorded and matched to corresponding events and times in the Centrax "User

Analysis Report". If an event was not shown in the report, then the actual audit log was inspected via Centrax's event viewer to see if the action triggered an audit log entry. Centrax did report all of the attacks, although careful examination and customization of the audit and detection policies was necessary to enable full protection.

This testing demonstrated many activities could generate false alarms in the audit trails. These alarms force the administrator to solve the problem by turning off the auditing, but that approach will leave an area unprotected. False alarms can be effectively filtered at one of three points: the audit policy, the detection policy, or the report generation. Filtering in the detection policy can significantly reduce the false positives from products such as backup tools and anti-virus scanners that open each file for modification during normal operation. Acceptable solutions to these types of problems will need to be researched and solved by both the software developer and the operating-system developer.

## E. SUMMARY OF AUDIT POLICY CHANGES DICTATED BY TESTING

Some specific observations from testing in the SPAWAR working environment:

- When all executable programs and their libraries are scanned and opened for write/append by the Norton Anti-Virus package, this causes large numbers of false alarms in the audit logs.
- The audit policy must check for any modifications by an administrator to personal-document directories and their contents. This will prevent administrator-account break-ins to get at critical Office documents in individual user's "My Document" directory.
- The audit policy must check for any modification to "cgi-bin" files and "htm" files, which make up the majority of Web page documents.
- A "Critical system files (modification)" signature should be added to the audit policy allowing for all critical system files to be audited when modified.
- The "single failed login" audit event was too sensitive for this environment, so this alert should be removed from the detection policy. The "three consecutive failed logins" audit event was retained.

- The file “win.ini” generates many false-alarms and should be removed from the audit policies
- The audit policy should check the application of user rights, which will detect sharing of directories.
- The audit and detection policy should check tampering with the login-banner bitmap file.
- The audit policy should be changed to include Restarts/Shutdowns to allow use of the “NT Starting” signature.
- The audit policy must include all *.dll and *.exe files.

## F. IT-21 LIMITATIONS

IT-21 was written considering the available NT audit tools that, up to NT Service Pack 4.0 with the addition of SCM, have been meager indeed. Without the advantage of commercial security tools specifically able to implement an audit policy across an entire network, any additions to the IT-21 audit policy could not have been implemented. As written, the IT-21 audit policy is only an NT audit capability guide, not a true security audit policy.

It is inappropriate to write a security-policy specification based only on available tools. While the IT-21 guide is an NT security implementation guide, without an NT security policy specification guide, the implementation guide by default dictates policy. Perhaps the fault lies not in the design of the implementation guide but in the lack of an NT security policy specification guide.

The limitations of the NT audit capabilities should not dictate audit policy. The size and the maintenance problems of audit files should not result in a tradeoff of size versus manageability and therefore limit auditing. The number of false alarms should not be the reason to turn off a segment of auditing but rather filtering rules should limit the number of false alarms. Another limitation on IT-21 audit policy is that there is no provision for reporting from the audit logs after the audit data has been collected.

The manual analysis of audit files has been successful in the past when only one system was being evaluated, but the continual growth and expansion of networks makes that no longer feasible. If better analysis tools are not developed to analyze and report on host-based intrusions and misuse, we might as well turn off all auditing altogether, for without automated analysis all we will see is what we have always seen from the auditing logs – highly unusual events that stand out. Such events will not appear for a trusted individual who is determined to compromise sensitive data and undermine the security of the military network.

## G. RECOMMENDED ADDITIONS TO IT-21 AUDIT POLICY

These are the additions to the IT-21 Audit Policy I recommend.

### 1. Login/Logoff Auditing

IT-21 login/logoff audit policy is to audit on success and failure. We recommend that particular emphasis must be placed on successful and unsuccessful logins to the NT default accounts Administrator and Guest; also any privileged accounts should be monitored. Filtering of successful logins is certainly needed. Login/logoff event types to monitor:

- Failed attempts to login as Administrator
- Failed attempts to login as Guest
- Failed login attempts to login as a user
- Failed logins to any privileged account
- Failed login attempts when account is locked out. (This setting of the permissible failed attempts to a low number such as three)
- Remote logins, including those over phone lines

### 2. File and Object Auditing

The IT-21 policy of only collecting file and object-access failures prevents the building of helpful rofiles. Object-access success should be collected too. File and object event types to monitor:

- Critical file modifications, additions, and deletions, especially those of system executables, script and command files, audit-log files, and web-page files
- Attempts to access files outside defined file space

- Access to the Critical System Registry
- Access to the Critical Project Registry
- Access to print jobs and the print queue

### 3. Use and User Rights Auditing

Use and user-rights auditing provides information on assignment of special privileges and use of those privileges. This information needs to be collected to provide possible clues to other event sequences that are problems but are not obviously suspicious.

### 4. User and Group Management Auditing

User and group management information needs to be collected to provide more clues to problems. User and Group Management event types to monitor:

- Any account changes including modifications, additions, and deletions, especially for privileged accounts and privileged group accounts
- Local and global group changes including group creation, member additions, member deletion, and group deletions

### 5. Security Policy Change Auditing

Auditing must especially cover implementation of security features throughout the operating system. All changes to the NT default security settings must be monitored for any type of access. Each audit event will need to be monitored for excessive false alarms, and filters may need to be designed.

## 6. Restart, Shutdown, and System Auditing

Restart, shutdown, and system auditing success/failure events should be collected.

## 7. Process Tracking

Process-tracking information collection can be used in statistical profile gathering.

## 8. Other Event Logs

Other event logs including the Application event log and the System event log should be monitored. The System event log should be monitored for system-device alerts including low disk and low main-memory space along with other system errors.

# IV. NT AUDIT SENSOR PROGRAM DESCRIPTION

## A. THE NT AUDIT SENSOR PROGRAM

Barrus [Ref. 1] proposed a system of independent agents on separate machines, working independently from each other, but cooperating by sending messages to each other regarding network intrusions. His Autonomous Agent design proposed that an agent monitor the activity on a single-host system in an attempt to identify single-host attacks as well as the components of network attacks that are occurring on that host. Ingram [Ref. 9] has implemented Barrus's design, incorporating sensors on each host throughout a network that send system-event information to agents (named IDAgents) on each host. In his system there is no central director or controller that could be compromised unlike with Centrax. Each IDAgent has its own controller and communications, and works with sensors specific to the logging systems they are interfaced to. The IDAgent receives input from its multiple sensors and informs other agents in the network of any unusual or anomalous activity. In this chapter I will describe the NT Audit Sensor, a sensor I built to interface the NT auditing system to the IDAgent. My experiments provide insight on alternatives to the Centrax approach to intrusion detection.

## B. NT EVENT LOG FILES

There are three Windows NT event logs. The Application event log records user-application events, both those selected by the application and system diagnostic events. All processes running under NT have the ability to log events to the application event log. The use of the application log by security programs needs to be expanded to include additional events. The System event log records events logged by Windows NT system services, drivers, and kernel mode events. The Security event log records Windows NT system security and system auditing events. The three event log files are located in the \WINNT\SYSTEM32\CONFIG directory. Each event log record is stored in the EVT binary record format and is only accessible using the Windows event-logging interface. We conducted some experiments with just the security event log.

The event log record fields are:

- The Time Generated field contains the date and time of the event measured in seconds since Jan. 1, 1970, at 00:00.
- The User is the user account associated with the event.
- The Computer Name is the computer on which the event occurred.
- The Event Identifier is the code number of reported event. This value is specific to the event source.
- The Event Source is the source name, application, service, driver, or subsystem that reported the event.
- The Event Type classifies the type of event. The Event Types of information, warning, and error are application-log types, while success-audit and failure-audit are security-log types.
- The Event Categories identifies logical categories to which this event belongs.
- The Message Strings describe the event in more detail.

## C. EVENT LOGGING FUNCTIONS

Twelve functions of the event logging interface [Ref. 14] can be used by application programs. OpenEventLog gets a handle to an already-open event log, opened by enabling auditing, while CloseEventLog closes it and releases its resources. ReadEventLog reads the specified event log. GetOldestEventLogRecord gets the index of the latest record and GetNumberOfEventLogRecords returns the number of records. RegisterEventSource returns a handle of a log file to be used by ReportEvent to write an event entry to an event log. This functionality allows a user program to report events to the event logging service for recording in the Application event log. DeregisterEventSource closes the log.

BackupEventLog copies the log before clearing to an archive; ClearEventLog clears all the events in a log file. OpenBackupEventLog opens a backup log file. NotifyChangeEventLog allows the logging service to signal to an event object created by CreateEvent, when a record is written to a specified log file. It allows the near real-time processing of security-event records as they are written, before they can be altered or deleted.

Here is an example of a security-event record [Ref. 15]. Event Identifier 539, an account being locked out, was created by three logins attempts to an account with an incorrect password when the set limit was two attempts. The NT Event Viewer program shows the record, which is an input to the NT Audit Sensor program.

| | | |
|---|---|---|
| Date | 07/15/1999 | |
| Time | 7:03:19 PM | |
| User | NT AUTHORITY\SYSTEM | |
| Computer | D871WS01 | |
| Event Identifier | 539 | ! 539 == Account Locked Out |
| (Event) Source | Security | |
| (Event) Type | Failure Audit | |
| (Event) Category | Login/Logoff | |
| Detail view: | | |
| Description: | *Event specific* | |
| Reason: | Account locked out | |
| User Name: | admin | ! Name of user |
| Domain: | D871WS01 | ! Domain (if server, else computer) |
| Login type: | 2 | ! 2 == local, 3 == remote |
| Login Process: | NWGINA | ! Process submitting login request |
| Authentication package: | MICROSOFT_AUTHENTICATION_PACKAGE_V1_0 | ! Program used to authenticate login |
| Workstation Name: | D871WS01 | |

My NT Audit Sensor program output fields are Date & Time Generated, Event ID, Event Type, Event Category, User Name, Computer name (Domain), Login Type, and Workstation. The format of the fields is ASCII text and they are comma-separated. The records are of variable length and fields reported are dependent on the record Event Identifier [Ref. 13]. An example record for the same event above is:

| | |
|---|---|
| Date | 7/15/1999 |
| Time | 19:03:19 PM |
| Event Identification | 539 |

| | | |
|---|---|---|
| Event Type | 16 | ! == Failure Audit |
| Event Category | 2 | ! == Login/Logoff |
| User | admin | |
| Computer | D871WS01 | !Domain |
| Login | 2 | !(local 2 or remote 3) |
| Workstation | D871WS01 | |

## D. NT AUDIT SENSOR PROGRAM SPECIFICS

My NT Audit Sensor program makes requests of the event logging service by calling the abovementioned functions to perform reading, backing up, and clearing of the security event log. The API (interface) Dynamic Link Library defines the functions and is used by user-mode Win32 processes to access the security-event log. The log can only be accessed by an user account with the "manage auditing and security log" privilege set. The event logging service can also access remote system' event logs by using Remote Procedure Calls but these calls were not explored in this program.

A block diagram of the sensor interface between my NT Audit Sensor and the IDAgent described in the thesis by Ingram [Ref. 9], is provided in the figure below.

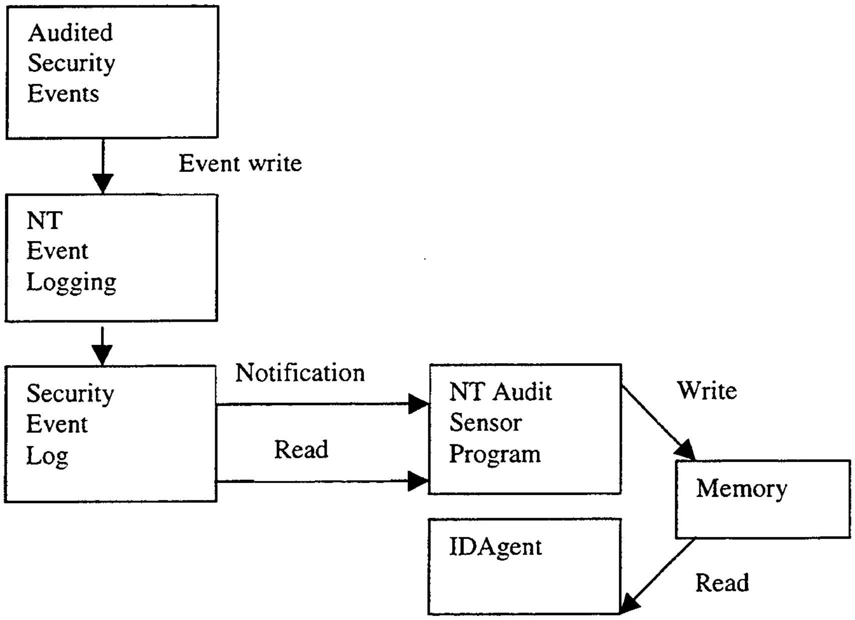

Figure 1. Context of the NT Audit Sensor Program

When important information is received from the sensor, the IDAgent's controller for that machine will construct a message and send it to other agents in the network to notify them of a security event or action.

The Audit Sensor program was written in C, using Microsoft Visual C++ 5.0. It was developed, compiled and linked using the Microsoft Visual Studio 97 environment. The executable program size is approximately 100 kilobytes.

The Main routine of the program uses the function NotifyThread to create a separate program thread to monitor the security event log and then wait for input. The NotifyThread routine was adapted from examples written by Murray [Ref. 18]. Within the thread, NotifyChangeEventLog associates the event file handle with an event object. The thread uses the WaitforSingleObject to detect when a new event log entry is written.

OpenEventLog then obtains a handle to the already open log. GetOldestEventLogRecord gets the index of the latest record and passes it to ReadEventLog to read the log. The Event Identifier finds the records of interest. The record is read in and reformatted to omit extraneous data, and is transferred to the IDAgent program via a pipe, which is a section of shared memory that processes use for communication. CloseEventLog closes the log and this releases the resources. The event object is then reset by ResetEvent to wait for the next event.

## E. TEST RUNS

The NT Audit Sensor program was tested on a dedicated NT workstation where NT security events of login success, login failure, and locked account occurred and were logged. The sensor program was notified through the NotifyChangeEventLog function when something was written to the NT security-event log, and if the event was of the types mentioned, it was parsed, reformatted, and written to a pipe. Initial standalone tests required the simulation of an IDAgent by a simple program that read the pipe and wrote to a disk file where the complete transfer could be checked.

The Sensor program provides real-time parsing of the security-event records to accomplish the following checks (admittedly incomplete) using Event Identification as the primary event-record key.

Detection: Audit Policy changed or Event Log cleared.

Description: Verify that Audit Policy changes reflect the organization policy and that the Event Log is only cleared by the Security Manager.

Action: Monitor the Security Event Log entry for Event Identification 517 where the user name is other than the Security Manager. Sensor provides a record for Event Identification 517, "Event Log cleared" from the first record to the new log file after the log has been cleared. Monitor the Security Event Log entries for Event Identification 612 where the user name is other than the Security Manager. Sensor provides the record for Event Identification 612, "Audit Policy Changed".

Detection: Attempt to exploit default NT user accounts, Administrator and Guest.

Description: The Administrator and Guest accounts that come with the initial NT system have been renamed by the security policy. Any Login Failure events with these account names indicate an attempted intrusion.

Action: Monitor the Security Event Log entries for Event Identification 529 where the account name is any spelling variation of account "Administrator" or "Guest". Sensor provides the record for Event Identification 529, "Login Failure".

Detection: Multiple Login Failures suggesting intrusion attempt.

Description: Multiple login failure events in a short period of time indicate a possible attempted intrusion. More weight should be given to a remote failure than a local failure.

Action: Monitor the Security Event Log entries for Event Identification 529. Compare the number from a workstation over a period of time. Sensor provides the record for Event Identification 529, "Login Failure".

Detection: Account lockouts suggesting intrusion attempt.

Description: Accounts are locked out when a defined number of unsuccessful login attempts occur. Attempts to login after lockout will result in additional lockout events. May indicate a possible attempted intrusion, but false alarms are possible.

Action: Monitor the Security Event Log entries for Event Identification 539 and Event Identification 644. Sensor provides the record for Event Identification 539, "Account locked" and for Event Identification 644, "User Account Locked".

Detection: User account added, deleted, or modified.

Description: User accounts are changed. May indicate a misuse, but false alarms are possible.

Action: Monitor the Security Event Log entries for Event Identifications 630, 624, and 642. Sensor provides the record for Event Identification 630, "User account delete", Event Identification 624, "User account created", and Event Identification 642, "User Account Modified".

The following is an output file, in the format described earlier, from an early test run of one complete pass of the Sensor program through a security event log. The log was interspersed with records of the type the program was capable of detecting and was created specifically for this test. All records of the types checked for were correctly identified.

```
8/31/99,17:43:46,612,8,6
8/31/99,16:52:30,624,8,7,test_u,D871WS01,admin
8/31/99,16:52:17,630,8,7,test_u,D871WS01,admin
8/31/99,16:50:42,612,8,6
8/31/99,16:49:19,624,8,7,test_user,D871WS01,admin
8/31/99,16:47:46,630,8,7,Test_acct_lockout,D871WS01,admin
8/31/99,16:47:24,612,8,6
8/31/99,15:14:52,529,16,2,admin,D871WS01,2,D871WS01
8/31/99,15:11:24,529,16,2,kremer,D871WS01,2,D871WS01
8/31/99,15:11:14,529,16,2,kremer,D871WS01,2,D871WS01
8/31/99,14:54:18,612,8,6
8/31/99,14:53:11,612,8,6
8/31/99,14:44:42,612,8,6
8/31/99,13:48:53,529,16,2,Administrator,D871WS01,2,D871WS01
8/31/99,13:48:42,529,16,2,Guest,D871WS01,2,D871WS01
```

```
8/31/99,13:28:3,539,16,2,Test_acct_lockout,D871WS01,2,D871WS01
8/31/99,13:28:1,529,16,2,Test_acct_lockout,D871WS01,2,D871WS01
8/31/99,13:28:1,644,8,7,Test_acct_lockout,D871WS01
8/31/99,13:27:58,529,16,2,Test_acct_lockout,D871WS01,2,D871WS01
8/30/99,8:20:50,529,16,2,ADMINTRJ,SPAWAR SSC ,3,\\FINA
8/26/99,15:18:52,529,16,2,admin,ANDROMEDA,2,D871WS01
8/26/99,15:18:39,529,16,2,admin,ANDROMEDA,2,D871WS01
8/26/99,15:18:27,529,16,2,admin,ANDROMEDA,2,D871WS01
8/26/99,14:57:46,517,8,1
```

The Audit Sensor program was then executed in a networked environment. The IDAgent, a software application running in background mode, as a Windows NT service, read the Sensor output record written to an output file rather than to a pipe, which we did not have the time to debug, and processed the information. The IDAgent is designed to take input from multiple sensors and send messages are sent to other IDAgents operating on other workstations and servers in the network. Ingram [Ref. 9] details the use of the Audit Sensor program and the results of the tests with the IDAgent. The sensor performed well, with no missed events, and the CPU time expended was minimal.

## F. POSSIBLE SENSOR PROGRAM UPGRADES

We suggest that this sensor program should be executed as a system service, started at system boot-up under NT System Services. It can be resident on all NT workstations and servers in the network. Since the security architecture and the auditing services are the same on servers and workstations, the sensor program can operate on both.

The sensor program currently recognizes only a small number of the events that can be audited, but it could be easily extended to handle more. Such events are set by the audit-policy settings which are mandated by the security policy. Programming to read the events and convert to a common format is needed, but the examples contained in the program should make that simple. A common audit format would allow data sharing amongst a variety of security tools.

The sensor could filter more false alarms if desired. A graphical user interface allowing selective filtering options would be helpful. The sensor monitors the security log only, but it could be adapted to monitor the application and system logs with additional threads.

To ensure that the sensor program is not sabotaged or crashed, a message indicating the sensor is alive should be send periodically to the IDAgent. If the IDAgent does not get the expected message, an appropriate response should be generated. File access auditing should be set for the sensor program so that any attempt to delete or take ownership of the sensor would be recorded.

Archive backup and clearing of the event logs are maintenance functions that could easily be added to the program. Examples of the functions to backup and clear the event logs are included in the program.

THIS PAGE IS INTENTIONALLY LEFT BLANK

# V. CONCLUSIONS AND RECOMMENDATIONS

With the ever-increasing size of the network environment, it becomes more and more important for the security manager to employ both host-based and networked intrusion-detection mechanisms. Along with the network size comes an ever-increasing volume of audit information, information that due to its sheer volume becomes unreviewable by a human. It is apparent that it is necessary to automate security monitoring tasks and to monitor activity in real time. Furthermore, only then can events affecting the entire enterprise be detected. Sites that do not observe up-to-date security practices risk intrusion from vulnerabilities of the NT operating system and security-management errors that will be exploited by persistent intruders.

## A. IT-21 AUDITING AND THIRD PARTY SECURITY PRODUCTS

The Navy IT-21 directive has gone a long way toward defining NT workstation and NT server security-installation requirements for C2 level in a cookbook fashion, but the auditing recommendations of IT-21 are limited, possibly based on the previous capabilities of NT audit-support tools. As an example of an IT-21 deficiency, there is no requirement in IT-21 to analyze the audit data in real time or to report on the data after collection. Audit data in more areas should be collected and real-time alarm and response mechanisms should be employed to analyze the audit data.

The Navy, under directives to integrate commercial off-the-shelf software, is looking at third-party security-software products to provide the level of security needed to operate in the current environment. To that end, CyberSafe was able to provide a partial solution to Microsoft's lack of audit-tool capability with their Centrax security product. But further developments are needed.

## B. NT AUDIT SENSOR

We also developed our own real-time intrusion sensor, the NT Audit Sensor. We demonstrated that it provides an efficient interface of the NT audit system an IDAgent distributed intrusion-detection system processing NT audit data in real-time and providing an alarm and a response if necessary. More audit data will need to be provided and more sensors will need to be developed to make the IDAgent useful in real-world settings. A standardized format is needed that would allow transparent interfacing to the IDAgent all operating systems and a mix of sensor types.

# APPENDIX: THE NT AUDIT SENSOR PROGRAM

/* The Sensor Main subroutine will create a thread which is set up to be notified when the Windows NT Security Event Log is written to. The sensor program can be set up to key on any Event Identification that the IDAgent decides on. The input is the most current record of the security event log.The Sensor output is a record with comma-separated fields, an ASCII null-terminated string transferred via a one-way write pipe to a second process, an IDAgent, an application program which make decisions based on a rule set. The rule set policy defines all of the detection rules. Each rule consists of clauses. The clauses define the conditions to monitor for, the conditions to ignore and the actions to take. The resultant action is based on a predetermined severity, and the IDAgent applies the appropriate response.
*/

```
/* Standard include files */
#include <windows.h>
#include <time.h>
#include <stdio.h>

/* NT Security Event Log */
#define EVENTLOG TEXT("SECURITY")

/* General String defines used for several different events */
#define User_Name          1
#define DomainName         2

/* Specific Event Id's to search for */
/* Defines for Login success/failure */
#define Login_success      528
#define Login_failure      529
/* Specific String definitions for Event ID 528, 529
 * defines Login Failure and Login Success          */
```

```
#define LocalRemote         3
#define Workstation         6

/* Defines for Account locked out */
#define Account_locked_out          539
#define User_account_locked_out     644
/* Specific String definitions for Event ID 539, 644
 * defines Account locked out where attempt is made to an
 * already locked account and user account locked out where
 * account is initially locked out       */
#define AccountName                 1
#define MachineName                 2

/* Defines for Account added/deleted */
#define Account_added               624
#define Account_deleted             630
/* Specific String definitions for Event ID 624, 630
 * defines Account deleted and Account added          */
#define UserAcctMGR                 4

/* Defines for Audit_policy_change and Audit_log_cleared */
#define Audit_policy_change         612
#define Audit_log_cleared           517

long WINAPI NotifyThread( LPARAM );
void SecLogFilehandling(void);

/* Time structure permits time to be formatted as we like it */
struct tm *ptm;
struct tm tm;
```

```
/* Global variables */
typedef unsigned long ulong;
HANDLE      hWrite;
HANDLE      hStdOut;

/*****
        Create pipe
*/
void create_pipe(PHANDLE phWrite)
{
 *phWrite = CreateNamedPipe("\\\\.\\pipe\\seclogevt",PIPE_ACCESS_OUTBOUND,
                PIPE_TYPE_MESSAGE | PIPE_READMODE_MESSAGE | PIPE_WAIT,
                1,0,0,0,NULL);
}

/*****
        Write to pipe
*/
void write_to_pipe(HANDLE hPipe, char *outbuf, ulong numchars)
{
ulong actual;

 WriteFile(hPipe, outbuf, numchars, &actual, FALSE);
}

/*****
        MAIN
*/
int main()
{
 DWORD  dwNotifyThreadId;
```

```
  HANDLE hNotifyThread;

  /* Create a thread to monitor for notifications */
  hNotifyThread = CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)NotifyThread,
                                         0, 0, &dwNotifyThreadId );

  /* if thread created then loop forever */
  if ( hNotifyThread )
  {
   /* Loop here forever */
   while ( 1 )
   {
               Sleep( 500 );
   }
  }
   return 0;
}

/*****
Create the thread and notify is Security Event Log is written to
*/
long WINAPI NotifyThread( LPARAM lParam )
{
 DWORD dwReturn = !NO_ERROR;
 HANDLE hSecLog;

 /* create the pipe */
 create_pipe(&hWrite);

 /* Open the Security event log on the local system */
 hSecLog = OpenEventLog( NULL, EVENTLOG );
```

```
  if ( hSecLog )
  {
   HANDLE hSecLogTrap;

/* Create an event to signal that the Security Log has had an entry written to it */
    hSecLogTrap = CreateEvent( NULL, TRUE, FALSE, NULL );
    if ( hSecLogTrap )
    {
      if ( NotifyChangeEventLog( hSecLog, hSecLogTrap ) )
      {
        /* Wait to be informed that the Security event log has changed */
        while ( hSecLogTrap )
        {
          if ( WaitForSingleObject( hSecLogTrap, INFINITE ) == WAIT_OBJECT_0 )
          {
            /* The security event log has had an event written to it */

/* Now handle reading new records and writing any parsed records to the pipe */
                        SecLogFilehandling();
            ResetEvent( hSecLogTrap );
          }
        }
      }
     CloseHandle( hSecLogTrap );
    }
   CloseEventLog( hSecLog );
  }
 ExitThread( dwReturn );
 return( dwReturn );
}
```

```
/*****
Security Log File Record Reading, Parsing and Write to Pipe.
*/
void SecLogFilehandling(void)
{
HANDLE hSecLogFile;
TCHAR  szBackupName[128];
DWORD dwBytesRead, dwBytesNeeded, dwReadFlags;
DWORD dwNumRecords;
DWORD dwNewestRecordIndex, dwOldestRecordIndex;
BYTE bBuffer[1024];

/* The next two lines are for debug to write to a file *
FILE *output_file;
char file_name [20] = "D:\\TEMP\\SECDATA.LOG"; * commented out */

/* Start of record header */
EVENTLOGRECORD *pRecord = (EVENTLOGRECORD *)bBuffer;

char   pipe_buffer[255], temp_buffer[255];
ulong  pipe_buffer_count;
ptm = &tm;     /* Point to the time structure - Time on record is in seconds
            from Epoch and needs to be reformatted */

/* Open the security event log on the local system */
if (hSecLogFile = OpenEventLog(NULL, EVENTLOG) )
 {
 /* Get the index to the newest record */
 GetOldestEventLogRecord(hSecLogFile, &dwOldestRecordIndex);
 GetNumberOfEventLogRecords(hSecLogFile, &dwNumRecords);
 dwNewestRecordIndex = (dwNumRecords + dwOldestRecordIndex) -1;
```

```
/*  Read security log backwards to get the most current record in the event
 log. When started with the system service process, the Sensor program
 reads the most current record and saves the time of that record. When
 new records are written to the Security Log, the program could read several
 records and compare against the saved time to identify new entries. */

 dwReadFlags = EVENTLOG_BACKWARDS_READ|EVENTLOG_SEEK_READ;
 pRecord->EventID =0;

 /* Read the latest Security Event Log record */
 ReadEventLog( hSecLogFile, dwReadFlags, dwNewestRecordIndex, &bBuffer,
sizeof(bBuffer),&dwBytesRead, &dwBytesNeeded );

 /* Multiple Event_ID's to be keyed on */
 if (((pRecord->EventID & 0x0000FFFF) == Account_locked_out) ||
   ((pRecord->EventID & 0x0000FFFF) == User_account_locked_out) ||
   ((pRecord->EventID & 0x0000FFFF) == Login_failure) ||
   ((pRecord->EventID & 0x0000FFFF) == Login_success) ||
   ((pRecord->EventID & 0x0000FFFF) == Account_deleted) ||
   ((pRecord->EventID & 0x0000FFFF) == Account_added) ||
   ((pRecord->EventID & 0x0000FFFF) == Audit_policy_change) ||
   ((pRecord->EventID & 0x0000FFFF) == Audit_log_cleared))
 {
  DWORD dwNumStrings, dwSizeData, dwSizeUsid;
  LPBYTE pStr;          /* Start of string data                    */
  LPBYTE pUsid;           /* Start of User Source IDentifier */
  LPBYTE pData;           /* Start of binary data                  */

  /* If security event record actually read */
  if (dwBytesRead > 0 )
```

```
        {
        /* Find start of Usid in the record */
        if ( dwSizeUsid = pRecord->UserSidLength )
          pUsid = (LPBYTE)pRecord + pRecord->UserSidOffset;

              /* Find start of variable Strings in the record */
        if ( dwNumStrings = pRecord->NumStrings )
          pStr = (LPBYTE)pRecord + pRecord->StringOffset;

              /* Find start of binary data in the record */
        if ( dwSizeData = pRecord->DataLength )
          pData = (LPBYTE)pRecord + pRecord->DataOffset;

              /* Convert the Epoch time to enable formatted time output */
              ptm = localtime(&pRecord->TimeGenerated);

              if (((pRecord->EventID & 0x0000FFFF) == Audit_policy_change) ||
                 ((pRecord->EventID & 0x0000FFFF) == Audit_log_cleared))
              {
  /* Building the standard record for output to the pipe */
pipe_buffer_count =
sprintf(pipe_buffer,"%d/%d/%d,%d:%d:%d,%lu,%lu,%lu\n",
                    ptm->tm_mon+1, ptm->tm_mday, ptm->tm_year, ptm->tm_hour,
                    ptm->tm_min, ptm->tm_sec, pRecord->EventID & 0x0000FFFF,
              pRecord->EventType, pRecord->EventCategory);
              }
       else  /* print Event ID, Type and Category to file */
              {
  /* Building the standard record for output to the pipe */
              pipe_buffer_count =
sprintf(pipe_buffer,"%d/%d/%d,%d:%d:%d,%lu,%lu,%lu,",
  ptm->tm_mon+1, ptm->tm_mday, ptm->tm_year, ptm->tm_hour,
  ptm->tm_min, ptm->tm_sec, pRecord->EventID & 0x0000FFFF,
  pRecord->EventType, pRecord->EventCategory);
  }

  /* parse out desired information from insertion strings of the
    security record. This portion of the record is specific for
    the Event ID */
  /* There is no aditional info captured for 517, and 612 */

  /* Specific for Event_ID = 528, 529, and 539 */
  if (((pRecord->EventID & 0x0000FFFF) == Login_failure) ||
     ((pRecord->EventID & 0x0000FFFF) == Login_success) ||
     ((pRecord->EventID & 0x0000FFFF) == Account_locked_out))
  {
   if ( pStr )
{
 DWORD i;
    for ( i = 0; i < dwNumStrings; i++ )
        {
        switch (i+1)
        {
         case User_Name:
          /* build record to pipe buffer */
          sprintf(temp_buffer,",%s,", pStr );
          strcat(pipe_buffer, temp_buffer);
          break;
         case DomainName:
          /* add on to record in pipe buffer */
          sprintf(temp_buffer,"%s,", pStr );
          strcat(pipe_buffer, temp_buffer);
```

```
                      break;
                    case LocalRemote:
                      /* add on to record in pipe buffer */
                      sprintf(temp_buffer,"%s,", pStr );
                      strcat(pipe_buffer, temp_buffer);
                      break;
                    case Workstation:
                      /* add on to record in pipe buffer */
                      /* remember this is end of record */
                      sprintf(temp_buffer,"%s", pStr );
                      strcat(pipe_buffer, temp_buffer);
                      break;
                 } /* end case statement */
                 /* Get the next string */
    pStr = strchr( pStr, '\0' ) + 1;
  } /* end for */
}  /* end if pStr */
    }    /* end if pRecord ... 528, 529 and 539 */
    /* Specific for Event_ID = 644 */
    else if ((pRecord->EventID & 0x0000FFFF) == User_account_locked_out )
    {
      if ( pStr )
{
 DWORD i;
 for ( i = 0; i < dwNumStrings; i++ )
                 {
                 switch (i+1)
                 {
                   case AccountName:
                     /* build record to pipe buffer */
                     sprintf(temp_buffer,",%s,", pStr );
```

```
              strcat(pipe_buffer, temp_buffer);
              break;
            case MachineName:
              /* add on to record in pipe buffer */
              sprintf(temp_buffer,"%s,", pStr );
              strcat(pipe_buffer, temp_buffer);
              break;
           } /* end case statement */
           /* Get the next string */
   pStr = strchr( pStr, '\0' ) + 1;
  } /* end for */
}   /* end if pStr */
   }     /* end else if pRecord ... 644 */
  /* Specific for Event_ID = 624, 630 */
  else if (((pRecord->EventID & 0x0000FFFF) == Account_added) ||
             ((pRecord->EventID & 0x0000FFFF) == Account_deleted ))
   {
    if ( pStr )
{
 DWORD i;
 for ( i = 0; i < dwNumStrings; i++ )
           {
           switch (i+1)
           {
            case AccountName:
             /* build record to pipe buffer */
             sprintf(temp_buffer,",%s,", pStr );
             strcat(pipe_buffer, temp_buffer);
             break;
            case DomainName:
             /* add on to record in pipe buffer */
```

```
                      sprintf(temp_buffer,"%s,", pStr );
                      strcat(pipe_buffer, temp_buffer);
                      break;
                    case UserAcctMGR:
                      /* remember this is end of record */
                      sprintf(temp_buffer,"%s,", pStr );
                      strcat(pipe_buffer, temp_buffer);
                      break;
                   }  /* end case statement */
                   /* Get the next string */
         pStr = strchr( pStr, '\0' ) + 1;
        } /* end for */
      }  /* end if pStr */
          }    /* end if pRecord ...for 624, 630 */

         /* now finish off the buffer and write pipe_buffer out to the pipe
         pipe_buffer_count = strlen(pipe_buffer);
    pipe_buffer[pipe_buffer_count] = '\n';
         pipe_buffer[pipe_buffer_count+1] = '\0';
         write_to_pipe(hWrite, pipe_buffer, pipe_buffer_count+1 ); */
    }  /* end if dwBytesRead > 0        */
  }    /* end if ReadEventLog = TRUE    */
}      /* end if OpenEventLog not NULL  */

sprintf( szBackupName, "%s.Backup", EVENTLOG );

/* Create a backup file of the security event log */
BackupEventLog( hSecLogFile, szBackupName );

/* Clear the Security Event Log if desired */
```

```
 /* Close the security event log */
 CloseEventLog( hSecLogFile );
}
```

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF REFERENCES

1. Barrus, Joseph D. "Intrusion Detection In Real Time In A Multi-Node, Multi-Host Environment", Naval Postgraduate School, September 1997.

2. Brinkley, Donald L. And Roger R. Schell. "What is there to worry about? (An Introduction to the Computer Security Problem)", *IEEE Software*, March 1993.

3. CINCPACFLT, Navy Administration Message: "InformationTechnology for the 21st Century," March 1977.

4. Cohen, Fred. "Information System Attacks: A Preliminary Classification Scheme", *Computers and Security*, 16:29-46, 1997.

5. Centrax Corporation, "Security Audit Training and Laboratory Support Final Report", Contract Number: GS-35F-4571G, January 1999.

6. Dept. of the Navy, Space and Naval Warfare Systems Command, Naval Information Systems Security Office, PMW 161. "Secure Windows NT Installation and Configuration Guide: Windows NT for Navy IT-21", Version 1.2, June 1998.

7. Dept. of the Navy, Space and Naval Warfare Systems Command, Naval Information Systems Security Office, PMW 161. "Secure Windows NT Installation and Configuration Guide, Service Pack 4,Windows NT for Navy IT-21", Version 1.4 , March 1999.

8. DOD 5200-28-STD Report, "Trusted Computer System Evaluation Criteria", Department of Defense, Washington, D.C., 1985.

9. Ingram, Dennis J. "Autonomous Agents for Distributed Intrusion Detection in a Multi-host Environment", Naval Postgraduate School, September 1999.

10. Kosoresow, Andrew P. and Steven A. Hofmeyr, "Intrusion Detection Via System Call Traces", *IEEE Software*, Sept.–Oct. 1997, pp 35-40.

11. Lunt, Theresa F. "A Survey of Intrusion Detection Techniques", *Computers & Security*, 12:405-418, 1993.

12. Mansour Esmaili, Rei Safavi-Naini, Bala Balachandran, Josef Pieprzyk. "Case-Based Reasoning for Intrusion Detection", *IEEE Software*, pp 214-223, 1996.

13. Microsoft Corporation. "Event Identifiers", http://msdn.microsoft.com/library/sdkdoc/winbase/eventlog_8uwj.htm, May 1999.

14. Microsoft Corporation. "Event Logging Functions, http://msdn.microsoft.com/library/sdkdoc/winbase/eventlog_8v53.htm, May 1999.

15. Microsoft Corporation. EVENTLOGRECORD", http://msdn.microsoft.com/library/sdkdoc/winbase/eventlog_7mcy.htm, May 1999.

16. Microsoft Corporation. "Event Types", http://msdn.microsoft.com/library/sdkdoc/winbase/eventlog_8v53.htm, May 1999.

17. Microsoft Corporation. "Introducing the NT 4.0 Service Pack 4" http://msdn.microsoft.com/library/periodic/period99/html/ewn9931.html, 1999.

18. Murray, James D. *Windows NT Event Logging*, O'Reilly, 1998.

19. Puketza, Nicholas, Mandy Ching, Ronald A. Olsson, and Biswanath Mukherjee. "A Software Platform for Testing Intrusion Detection Systems", *IEEE Software*, pp 43-51, Sept.-Oct. 1997.

20. Staniford-Chen, Stuart, Brian Tung, and Dan Schnackenberg. "The Common Intrusion Detection Framework (CIDF)", http://seclab.cs.ucdavis.edu/cidf/papers/isw.txt, August 30, 1999.

21. Tapper, Daniel R. "A Comparison of the Security Architectures of Microsoft Windows NT 4.0 and Novell IntranetWare 4.11", Microsoft Corporation, 1998.

22. The Burton Group. "Network Strategy Report – Windows NT Security", V1, h23tp://www.tbg.com/samples.netsvcs/winntsc.htm, March 6, 1998.

# BIBLIOGRAPHY

The following lists several resources that were very helpful in preparing this thesis, but were not actually cited in the text.

Kayas, Othmar. *Internet Security: Risk Analysis. Strategies and Firewalls*, International Thompson Publishing, 1997.

Kaufman, Perlman, and Spencer, *Network Security*, Prentice-Hall, 1996.

Microsoft Corporation. "Event Data", http://msdn.microsoft.com/library/sdkdoc/winbase/eventlog_3uxt.htm, May 1999.

Microsoft Corporation. "Querying the Event Log", http://msdn.microsoft.com/library/sdkdoc/winbase/eventlog_6b6v.htm, August 1999.

THIS PAGE INTENTIONALLY LEFT BLANK

## INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center..........................................................2
8725 John J. Kingman Road, Ste 0944
Ft. Belvoir, Virginia 22060-6218

2. Dudley Knox Library ..........................................................................2
Naval Postgraduate School
411 Dyer Rd.
Monterey, California 93943-5101

3. Chairman, Code CS.............................................................................1
Computer Science Department
Naval Postgraduate School
Monterey, CA 93940-5000

4. Dr. Neil Rowe..................................................................................1
Computer Science Department, Code CS/Rp
Naval Postgraduate School
Monterey, California 93943-5100